

6. 자바 라이브러리

진짜 닷컴 게임을 만듭니다.

ArrayList에 대해 알아봅니다.

자바 API를 활용하는 방법에 대해 알아봅니다.

자바 API를 알아봅시다



정말인가? 우리가 직접
만들지 않아도 되는 건가?

SimpleStartupGame의 버그

정상적인 실행 결과

```
$ java SimpleStartupGame
enter a number      1
miss
enter a number      2
miss
enter a number      3
miss
enter a number      4
hit
enter a number      5
hit
enter a number      6
kill
6 guesses
```

버그가 나는 경우

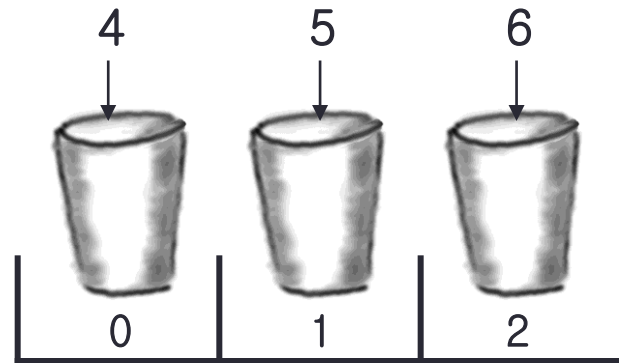
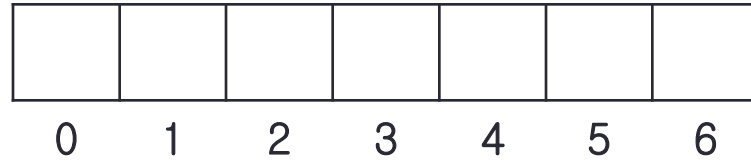
```
$ java SimpleStartupGame
enter a number      2
hit
enter a number      2
hit
enter a number      2
kill
3 guesses
```

일단 어떤 숫자를 한 번 맞추고 나면 그 숫자를 두 번 더 입력하기만 하면 게임이 끝납니다.

문제의 원인

```
public String checkYourself(int guess) {  
    // int guess = Integer.parseInt(stringGuess);  
    String result = "miss";  
    for (int i = 0; i < locationCells.length; i++) {  
        if (guess == locationCells[i]) {  
            result = "hit";  
            numOfHits++;  
            break;  
        }  
    }  
    if (numOfHits == locationCells.length) {  
        result = "kill";  
    }  
    System.out.println(result);  
    return result;  
}
```

어떻게 고쳐야 할까요?

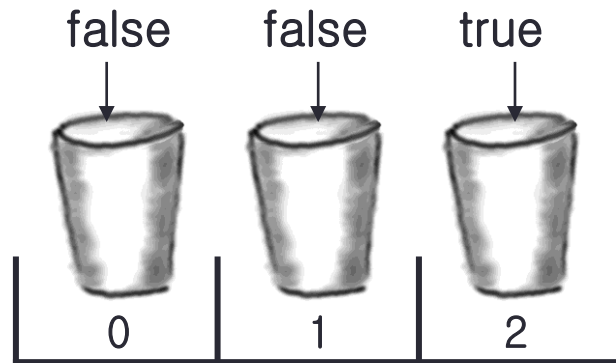


locationCells
(Startup의 인스턴스 변수)

Startup의 셀 위치를 저장하기 위한 배열 인스턴스 변수. 이 닷컴에는 4, 5, 6이라는 세 값이 들어있고 사용자는 이 세 개의 숫자를 맞춰야 합니다.

고치는 방법 - 첫 번째 옵션

배열을 하나 더 만들고 사용자가 위치를 맞출 때마다 그 맞춘 위치를 두 번째 배열에 집어넣습니다. 사용자가 위치를 입력할 때마다 그 배열을 확인하여 그 셀을 이미 맞췄는지 확인합니다.



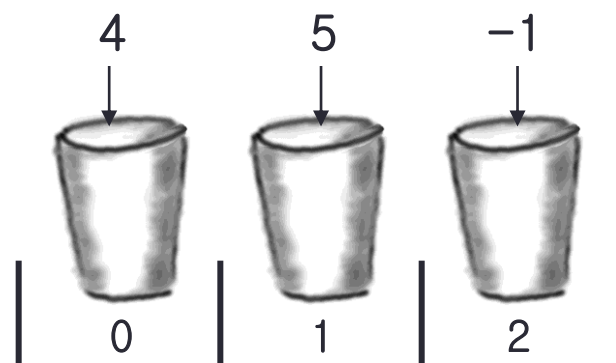
hitCells
(Startup의 인스턴스 변수)

```
boolean allChecked = true;
for (boolean r : hitCells) {
    allChecked &&= r;
}
if (allChecked) { }
```

단점: 작업이 복잡해집니다.

고치는 방법 - 두 번째 옵션

원래 배열을 그냥 사용하면서 맞춘 셀의 값은 -1로 바꿉니다.

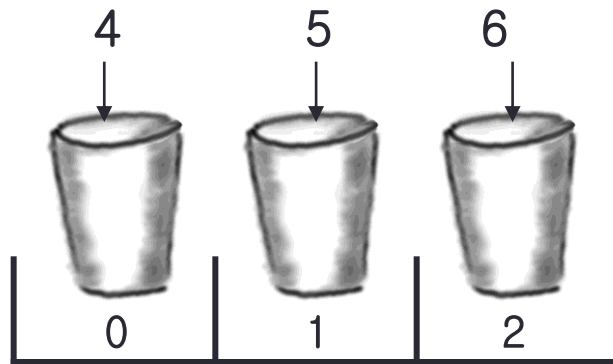


locationCells
(Startup의 인스턴스 변수)

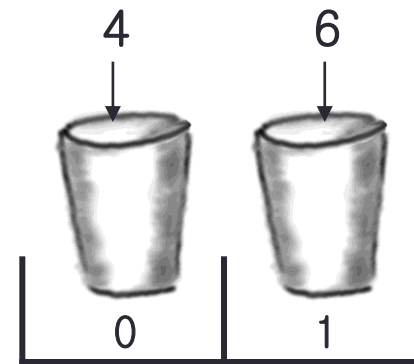
장점: 배열을 추가할 필요가 없습니다.
단점: 여전히 지지분합니다.

고치는 방법 – 세 번째 옵션

맞출 때마다 각 셀 위치를 삭제하고 배열을 작게 줄입니다.
하지만 배열의 크기를 바꿀 수는 없기 때문에 매번 새로운 배열을 만
들고 이전 배열에 남아있는 셀을 복사하고 새로 만든 작은 배열로 옮
겨야 합니다.



아무 셀도 맞추지 않은 상태



1번 인덱스에 들어있는 '5'를 맞춘 상태

장점: 아주 이상적인 해결책입니다.

단점: 배열의 크기를 바꿀 수 있어야 깔끔합니다.

checkYourself 준비 코드

```
메서드: String checkYourself(String userGuess)
사용자가 추측한 위치를 String 매개변수로 받아옴
사용자가 추측한 위치를 int로 변환
int 배열의 각 셀에 대해 다음 작업 반복
// 사용자가 추측한 위치를 셀과 비교
만약 사용자가 추측한 것이 맞으면
    맞춘 개수 증가
// 마지막 위치인지 확인
만약 맞춘 회수가 3이면 kill을 리턴
그렇지 않으면 hit 리턴
만약 부분 끝
그렇지 않으면 miss 리턴
만약 부분 끝
반복 부분 끝
메서드 끝
```

```
메서드: String checkYourself(String userGuess)
사용자가 추측한 위치를 String 매개변수로 받아옴
사용자가 추측한 위치를 int로 변환
남아있는 각 위치 셀에 대해 다음 작업 반복
// 사용자가 추측한 위치를 셀과 비교
만약 사용자가 추측한 것이 맞으면
    그 셀을 배열에서 제거
// 마지막 위치인지 확인
만약 배열이 비어 있으면 "kill"을 리턴
그렇지 않으면 "hit" 리턴
만약 부분 끝
그렇지 않으면 "miss" 리턴
만약 부분 끝
반복 부분 끝
메서드 끝
```

라이브러리

- 자바에는 미리 만들어진 클래스가 엄청나게 많이 있습니다.
- 이렇게 미리 만들어져 있는 클래스는 이 책에 나오는 인스턴트 코드랑 비슷하다고 보면 되는데 모두 미리 컴파일이 된 상태로 제공됩니다.
- 라이브러리에 들어있는 클래스는 필요에 맞게 잘 찾아서 쓰기만 하면 됩니다.

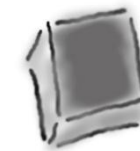
java.util.ArrayList

- **add(Object o)**
 - 객체 매개변수(o)를 목록에 추가
- **remove(int index)**
 - index 매개변수로 지정한 위치에 있는 객체를 제거
- **remove(Object elem)**
 - 주어진 객체가 ArrayList에 들어있으면 제거
- **contains(Object elem)**
 - 객체 매개변수 elem에 매치되는 것이 있으면 true 리턴
- **isEmpty()**
 - 목록에 아무 원소도 없으면 true 리턴
- **indexOf(Object elem)**
 - 객체 매개변수(elem)의 인덱스 또는 -1을 리턴
- **size()**
 - 현재 목록에 들어있는 원소의 개수 리턴
- **get(int index)**
 - 주어진 index 매개변수 위치에 있는 객체를 리턴

ArrayList로 할 수 있는 것

1. 새로 만듭니다.

```
ArrayList<Egg> myList = new ArrayList<Egg>();
```



2. 뭔가를 집어넣습니다.

```
Egg egg1 = new Egg();
```

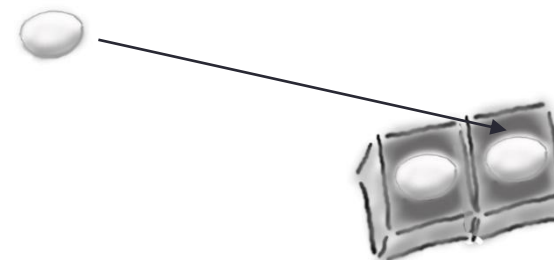
```
myList.add(egg1);
```



3. 다른 것을 또 집어넣습니다.

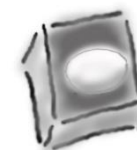
```
Egg egg2 = new Egg();
```

```
myList.add(egg2);
```



ArrayList로 할 수 있는 것

4. 몇 개가 들어가 있는지 알아냅니다.
`int theSize = myList.size();`
5. 어떤 것이 안에 들어있는지 찾아냅니다.
`boolean isIn = myList.contains(egg1);`
6. 어떤 것의 위치(인덱스)를 찾아냅니다.
`int idx = myList.indexOf(egg2);`
7. 비어있는지 확인합니다.
`boolean empty = myList.isEmpty();`
8. 목록에 들어있는 것을 제거합니다.
`myList.remove(s);`



Q&A

- ArrayList가 참 좋은 건 알겠는데, 그런 게 있다는 것을 어떻게 알아내죠?
 - 그런 걸 알아내는 능력이 자바 프로그래머로 성공하는 데 있어서 가장 중요한 것 가운데 하나라고 할 수 있습니다. **필요한 것을 찾아내서 적용할 수 있어야만 하지요.** 귀찮고 자질구레한 부분은 남들이 만들어놓은 것을 잘 활용하고 자신의 아이디어만 구현하는 “게으름의 미덕”을 충분히 활용할 수 있어야 합니다.
- **참고: devdocs, java 21 api 를 찾아보세요~**

- ArrayList라는 것이 필요하다는 것은 어떻게 알아내죠? API를 어떻게 활용할지를 알아내는 방법이 궁금합니다.
- 이 강의가 끝날 무렵에는 자바라는 언어에 대해서는 잘 알 수 있을 것입니다. 중요한 것은 어떤 문제를 어떻게 해결할지, 그리고 어떻게 하면 최소한의 코드만으로 해결할 수 있는가 하는 것입니다. 잠시 후에 이와 관련된 이야기가 나올 것입니다. 조금만 기다려 주세요.

ArrayList와 일반 배열

ArrayList

일반 배열

<code>ArrayList<String> myList = new ArrayList<String>();</code>	<code>String[] myList = new String[2];</code>
<code>String a = new String("whoohoo");</code>	<code>String a = "whoohoo";</code>
<code>myList.add(a);</code>	<code>myList[0] = a;</code>
<code>String b = new String("Frog");</code>	<code>String b = new String("Frog");</code>
<code>myList.add(b);</code>	<code>myList[1] = b;</code>
<code>int theSize = myList.size();</code>	<code>int theSize = myList.length;</code>

ArrayList와 일반 배열

ArrayList

일반 배열

<code>String str = myList.get(1);</code>	<code>String str = myList[1];</code>
<code>myList.remove(1);</code>	<code>myList[1] = null;</code>
<code>boolean isln = myList.contains(b);</code>	<code>boolean isln = false;</code> <code>for (int i = 0; i < myList.length; i++) {</code> <code>if (b.equals(myList[i])) {</code> <code>isln = true;</code> <code>break;</code> <code>}</code> <code>}</code>

ArrayList와 일반 배열

1. 일반 배열은 만들어질 때부터 유형과 크기가 정해집니다.
 - 하지만 ArrayList에서는 ArrayList 유형의 객체만 만들면 됩니다. 객체를 추가하거나 제거할 때마다 저절로 커지거나 작아지기 때문에 따로 크기를 지정할 필요가 없습니다.

2. 객체를 일반 배열에 집어넣을 때는 위치를 지정해야 합니다.

- 인덱스가 배열의 끝을 넘어가면 큰 문제가 생길 수 있습니다. 하지만 ArrayList를 쓸 때는 `add(anObject)` 같은 식으로 객체를 추가하기만 하면 새로 추가된 객체가 들어갈 자리를 알아서 만들어줍니다.

3.배열에서는 다른 어떤 자바 객체에서도 쓰지 않는 문법을 사용합니다.

- 배열에서는 myList[1]과 같은 특별한 문법을 쓰지만 ArrayList는 일반 자바 객체이므로 그러한 특별한 문법을 익히지 않아도 됩니다.

4. 자바 5.0에는 ArrayList가 매개변수화되어 있습니다.

- 자바 5.0에서는 ArrayList에 **매개변수화된 유형(parametrized type)**이라는 특별한 기능이 추가되었습니다.
- **ArrayList<String>**에서 String은 매개변수 유형입니다.
- **특정 유형의 객체들만 저장할 수 있는 컬렉션**을 만들 수 있습니다.

기존 코드



개선한 코드

```
public class Startup {
    int[] locationCells;
    int numOfHits = 0;

    public void setLocationCells(int[] locs) {
        locationCells = locs;
    }

    public String checkYourself(String stringGuess) {
        int guess = Integer.parseInt(stringGuess);
        String result = "miss";
        for (int cell : locationCells) {
            if (guess == cell) {
                result = "hit";
                numOfHits++;
                break;
            }
        }
        if (numOfHits == locationCells.length) {
            result = "kill";
        }
        System.out.println(result);
        return result;
    }
}
```

pp.180-181

```
import java.util.ArrayList;

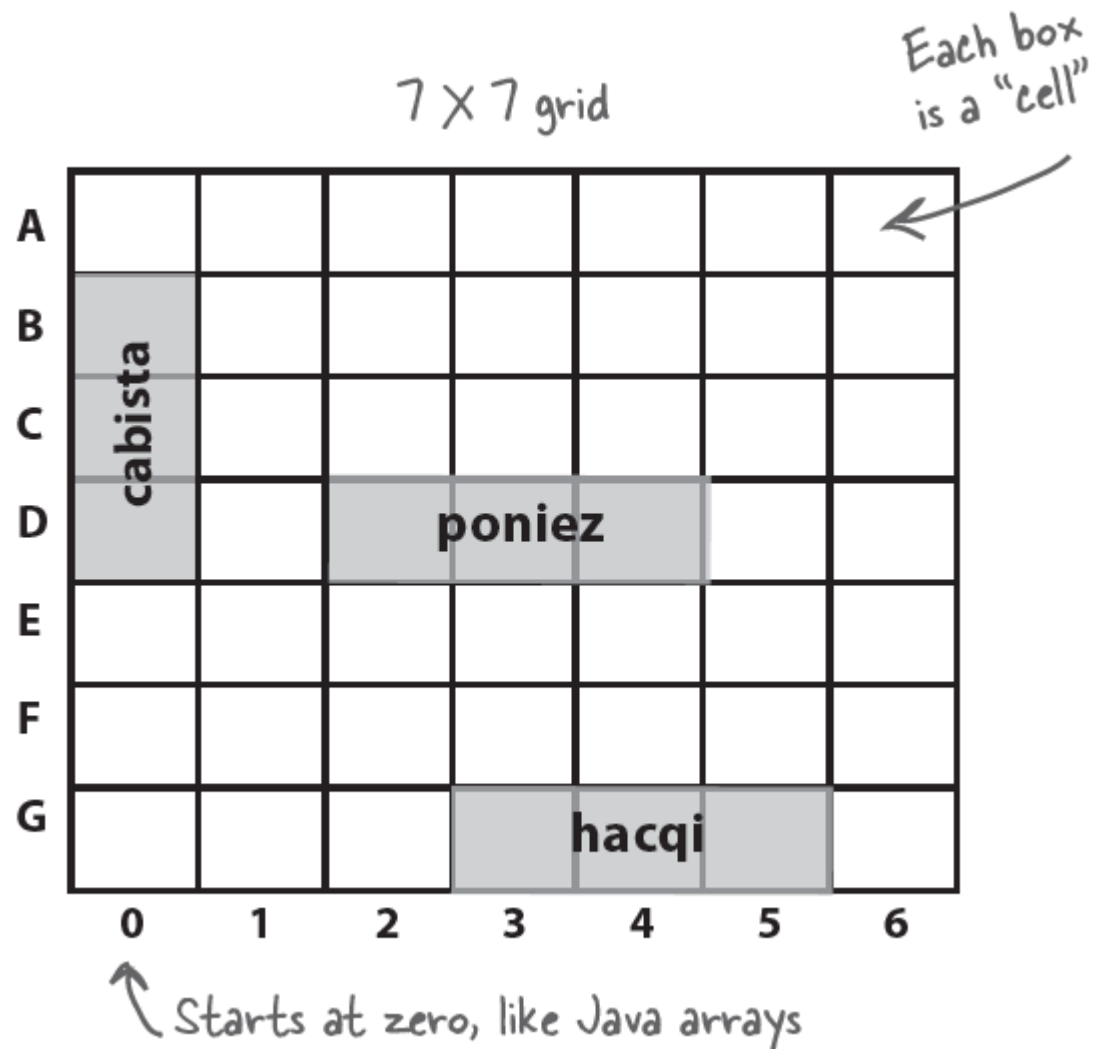
public class Startup {
    private ArrayList<String> locationCells;
    // private int numOfHits;

    public void setLocationCells(ArrayList<String> loc) {
        locationCells = loc;
    }

    public String checkYourself(String userInput) {
        String result = "miss";
        int index = locationCells.indexOf(userInput);
        if (index >= 0) {
            locationCells.remove(index);
            if (locationCells.isEmpty()) {
                result = "kill";
            } else {
                result = "hit";
            }
        }
        return result;
    }
}
```



진짜 닷컴 가라앉히기 게임



```
$ java StartupBust
Enter a guess A3
miss
Enter a guess B2
miss
Enter a guess C4
miss
Enter a guess D2
hit
Enter a guess D3
hit
Enter a guess A3
Ouch! You sunk poniez :(
kill
Enter a guess B4
miss
Enter a guess G3
hit
Enter a guess G4
hit
Enter a guess G5
Ouch! You sunk hacqi :(
```

고쳐야 할 부분

- Startup 클래스 (닷컴 클래스)
 - name 변수 추가
 - 닷컴의 이름을 저장하기 위한 변수. 닷컴이 죽었을 때 그 이름을 출력합니다.
- StartupBust 클래스 (게임 클래스)
 - Startup 객체 세 개 만들기
 - **세 닷컴을 7x7 그리드에 배치**
 - 세 닷컴에 대해 사용자가 추측한 위치 확인
 - 모든 닷컴이 가라앉을 때까지 게임 진행
 - main 메서드 종료

필요한 클래스와 객체

StartupBust	Startup	GameHelper
게임 클래스 Startup을 만들고 사용자의 입력을 받아들이고 모든 닌컴이 죽을 때까지 계속 진행	실제 Startup 객체 닌컴명, 위치, 사용자가 추측한 위치가 맞는지 확인하는 방법 등을 제공	보조 클래스 사용자가 명령행에 입력한 것을 받아오는 방법, Startup의 위치를 만들어내는 방법 등을 제공

5 Objects:



StartupBust



Startup
Startup
Startup

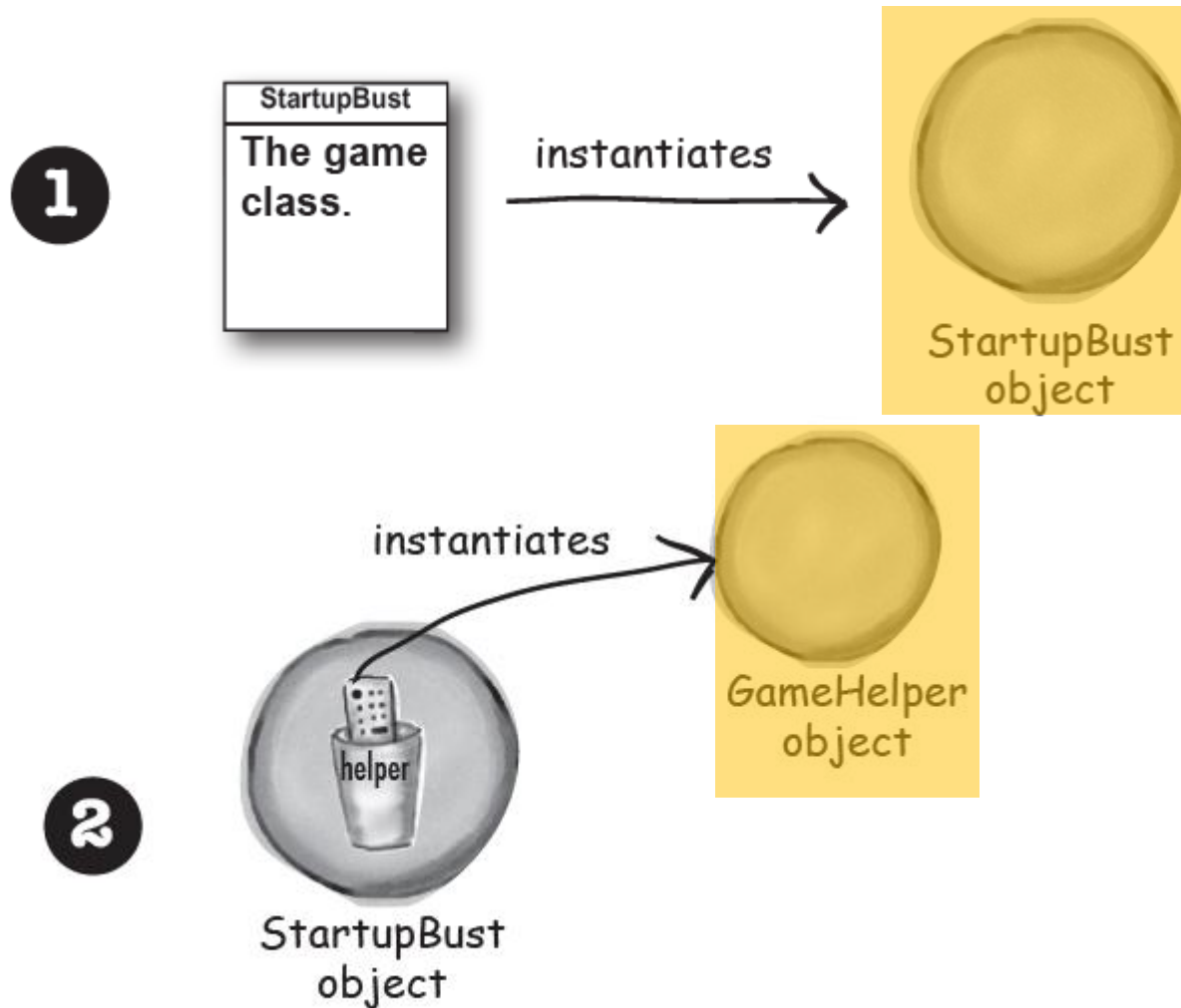


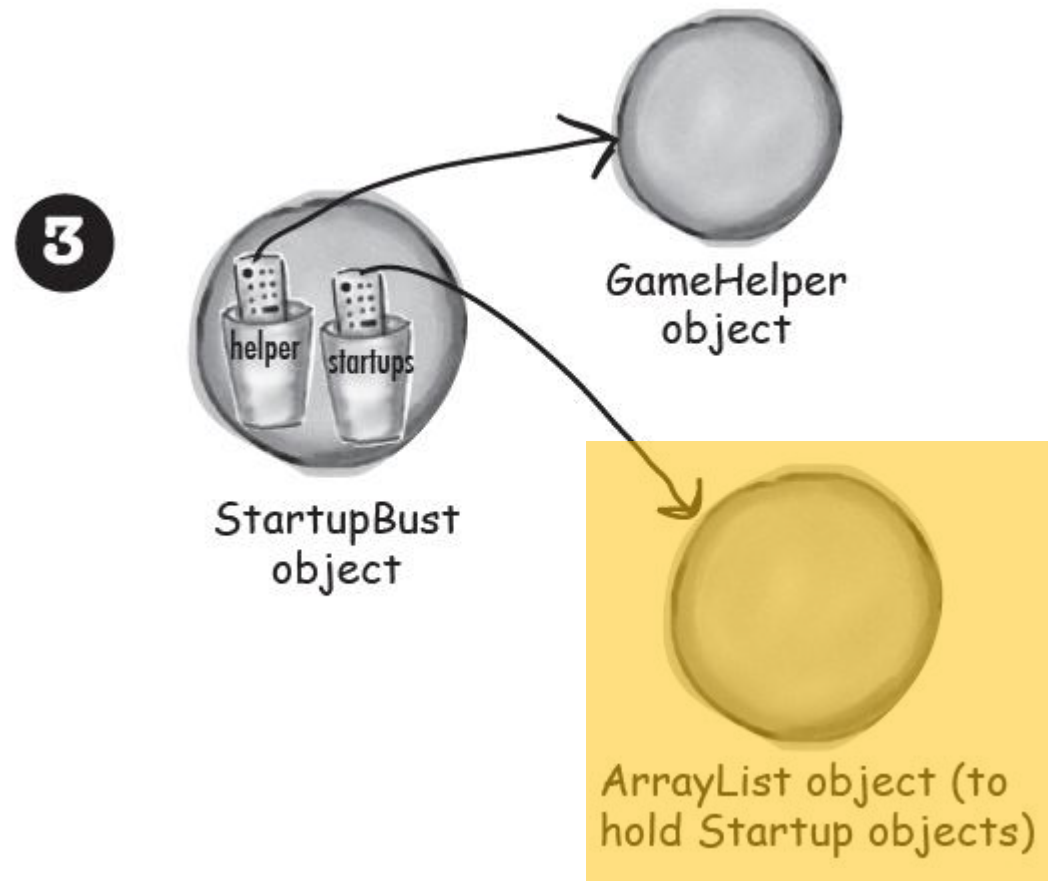
GameHelper

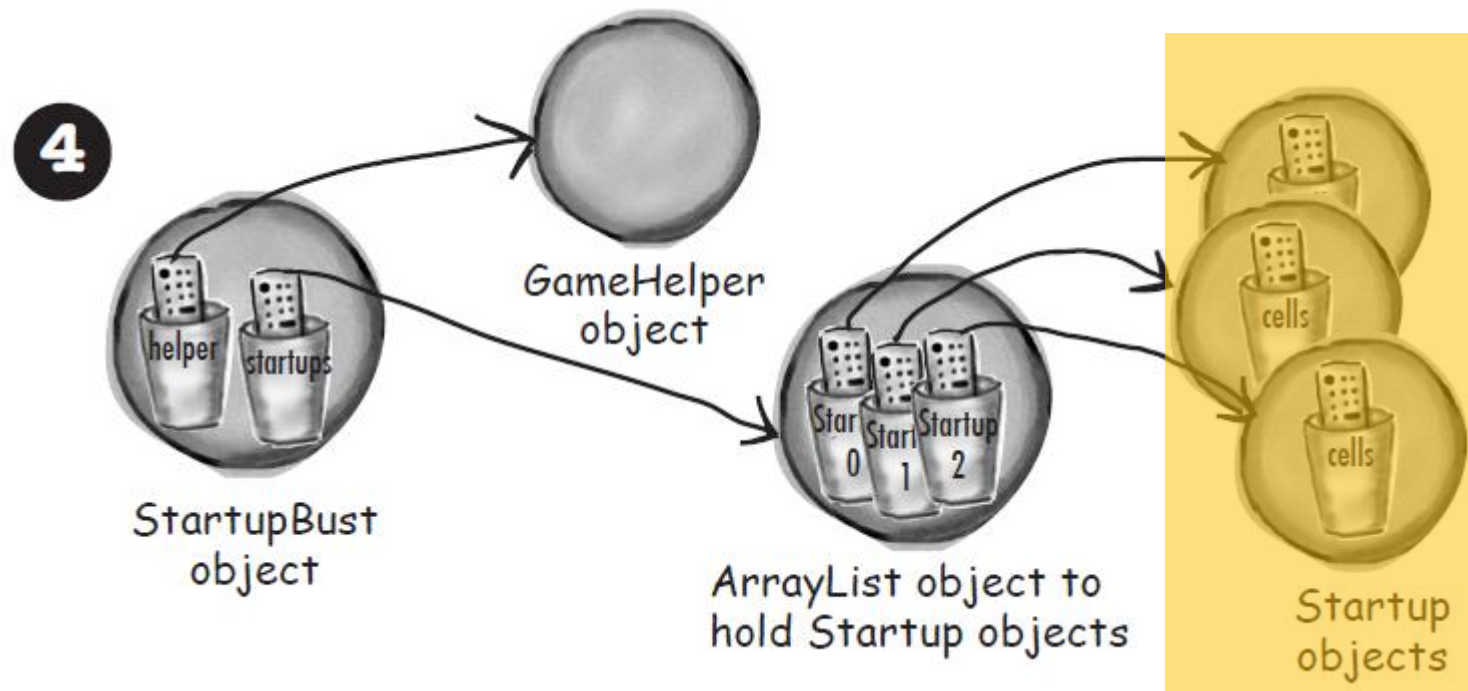


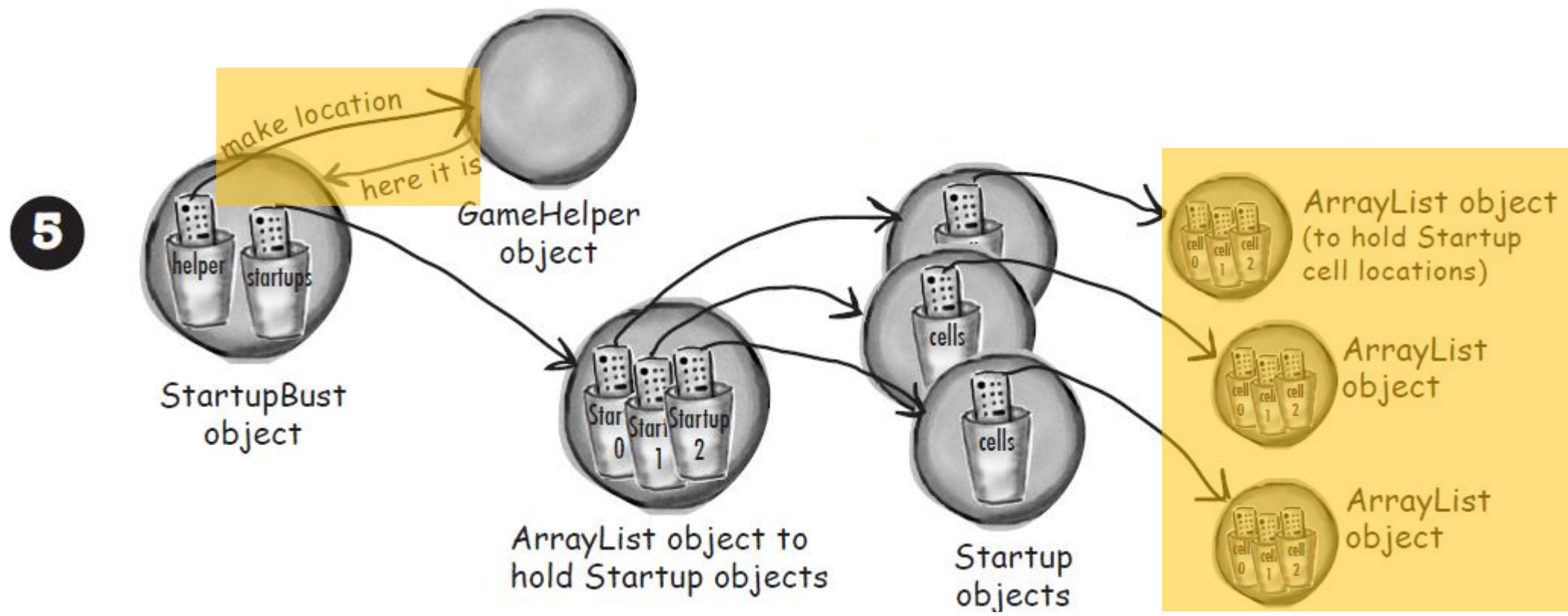
ArrayList
ArrayList
ArrayList
ArrayList

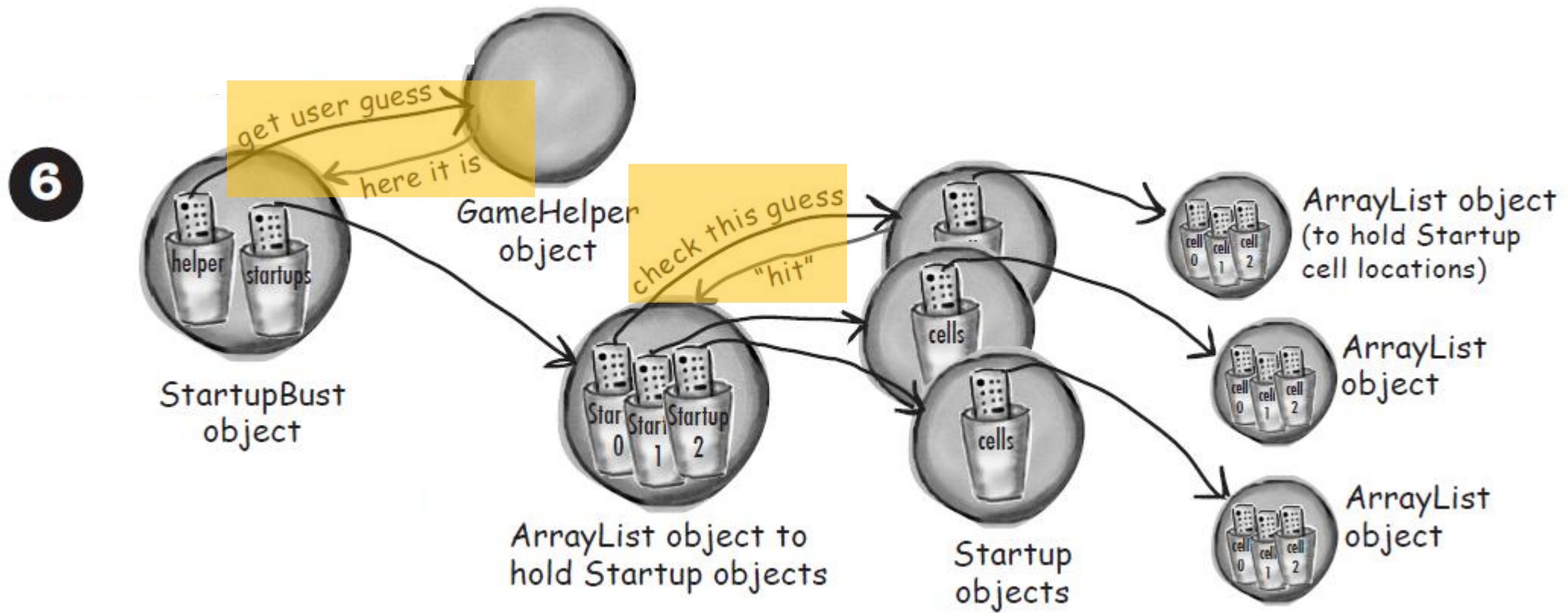
StartupBust 객체에서 일어나는 일











StartupBust 준비 코드

StartupBust
GameHelper helper ArrayList StartupsList int numOfGuesses
setUpGame() startPlaying() checkUserGuess() finishGame()

GameHelper 인스턴스를 만들고 helper라는 인스턴스 변수 선언
Startup 목록 세 개를 저장하기 위한 StartupsList라는 ArrayList 선언
추측 회수를 저장하기 위한 numOfGuesses라는 int 선언, 0으로 설정

setUpGame() 메서드 선언
startPlaying() 메서드 선언
finishGame() 메서드 선언

메서드: void setUpGame()

// Startup 객체 세 개 만들고 이름을 붙임

Startup 객체 세 개를 생성

각 Startup의 이름을 설정

모든 Startup을 StartupsList라는 ArrayList에 추가

StartupsList에 들어있는 각 Startup 객체에 대해 다음을 반복

 helper 객체의 placeStartup() 메서드 호출

 placeStartup()을 호출한 결과를 바탕으로 각 Startup의 위치 설정

반복 끝

메서드 끝

StartupBust 준비 코드

메서드: void startPlaying()

Startup이 남아있는 동안 다음을 반복

helper의 getUserInput() 메서드 호출, 사용자로부터 입력을 받음

checkUserGuess() 메서드로 사용자가 추측한 위치를 평가

반복 끝

메서드 끝

메서드: void checkUserGuess(String userGuess)

// 명중된 Startup 객체가 있는지 확인

numOfGuesses 변수에 저장된 추측 회수 증가

result라는 지역 변수를 “miss”로 설정

StartupsList에 들어있는 각 Startup 객체에 대해 다음을 반복

Startup 객체의 checkYourself() 메서드로 추측한 위치 평가

리턴값에 따라 result의 값을 “hit” 또는 “kill”로 설정

만약 결과가 “kill”이면 그 Startup을 StartupsList에서 제거

반복 끝

result 값 출력

메서드 끝

StartupBust 준비 코드

메서드: **void finishGame()**

게임이 끝났음을 알리는 메시지를 출력

만약 추측한 회수가 적으면

축하 메시지를 출력

그렇지 않으면

결과가 좋지 않다는 메시지를 출력

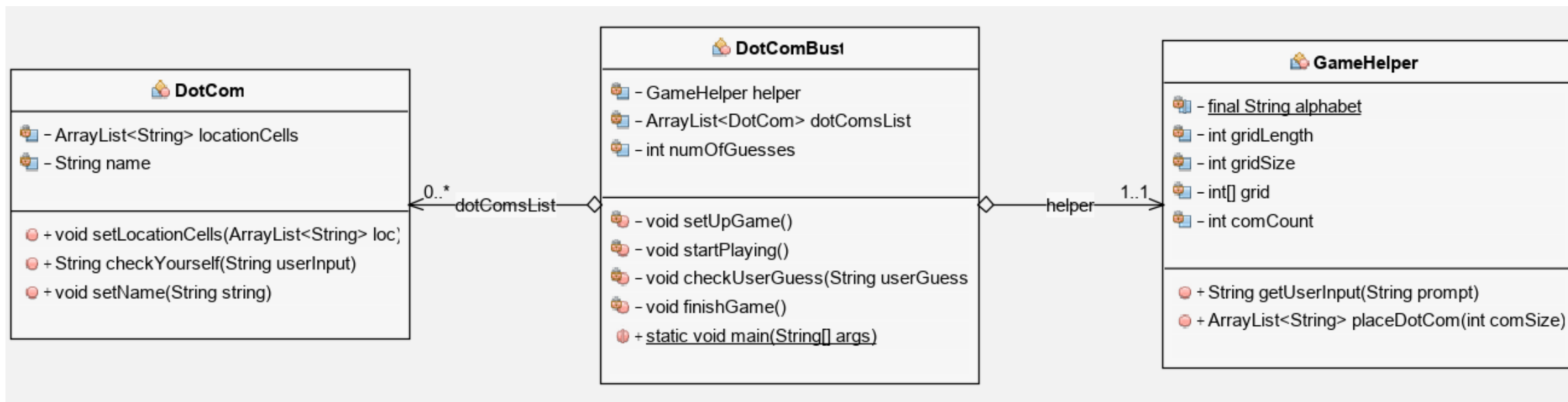
만약 부분 끝

메서드 끝

여기에서는 테스트 코드를 만들고 있지 않지만, 실제로 프로그래밍을 할 때는 테스트 코드를 만드는 것이 좋습니다. → test-first programming

각 메서드에 대해 테스트 코드를 만들면서 그 테스트를 통과할 수 있는 메서드(실제 코드)를 만들고, 다시 테스트 코드를 수정하고 그 테스트 코드를 통과할 수 있는 메서드 실제 코드를 만드는 식의 작업을 반복하다 보면 실제 코드가 완성이 되겠죠?

클래스 다이어그램



DotCom ⇔ Startup 로 대체함.

6장 소스 코드 (2판)

- cse.oop2.ch06 패키지
 - DotCom.java
 - GameHelper.java
 - DotComBust.java

DotCom.java

예) userInput = "g0"

```

1  package cse.oop2.ch06;
2
3  import java.util.ArrayList;
4
5  public class DotCom {
6
7      private ArrayList<String> locationCells;
8      private String name; // 새로 추가
9
10     public void setLocationCells(ArrayList<String> loc) {
11         locationCells = loc;
12     }

```

예) loc = [b0, c0, d0]

System.out.println("loc = " + loc);

```

14     public String checkYourself(String userInput) {
15         String result = "miss";
16         int index = locationCells.indexOf(userInput);
17         if (index >= 0) {
18             locationCells.remove(index);
19             if (locationCells.isEmpty()) {
20                 result = "kill";
21             } else {
22                 result = "hit";
23             }
24         }
25         return result;
26     }
27
28     public void setName(String string) {
29         name = string;
30     }
31 }

```

GameHelper.java

```

1  package cse.oop2.ch06;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.util.ArrayList;
7
8  public class GameHelper {
9
10     private static final String alphabet = "abcdefg";
11     private final int gridLength = 7;
12     // 7x7 2차원 아니라 1차원임.
13     private int gridSize = 49;
14     private final int[] grid = new int[gridSize];
15     private int comCount = 0;
16
17     public String getUserInput(String prompt) {
18         String inputLine = null;
19         System.out.print(prompt + " ");
    
```

값이 더 이상 변하지 않으면 final로 지정. class, method에도 final 지정 가능!

49개 정수 모두 0으로 초기화

세 개의 Startup
객체를 자동으로
배열하는 메서드

```

20     try {
21         BufferedReader is = new BufferedReader(
22             new InputStreamReader(System.in));
23         inputLine = is.readLine();
24         if (inputLine.length() == 0) {
25             return null;
26         }
27     } catch (IOException e) {
28         System.out.println("IOException: " + e);
29     }
30     return inputLine.toLowerCase();
31 }
32
33 public ArrayList<String> placeDotCom(int comSize) { // line 19
34     ArrayList<String> alphaCells = new ArrayList<String>();
35     String[] alphacoords = new String[comSize]; // holds 'f6' type coords
36     String temp = null; // temporary String for concat
37     int[] coords = new int[comSize]; // current candidate coords
38     int attempts = 0; // current attempts counter
39     boolean success = false; // flag = found a good location ?
40     int location = 0; // current starting location

```

comSize = 3

NOT USED

```

42 comCount++; // nth dot com to place
43 int incr = 1; // set horizontal increment
44 if ((comCount % 2) == 1) {
45     incr = gridLength; // set vertical increment
46 }
47
48 while (!success & attempts++ < 200) {
49     // get random starting point
50     location = (int) (Math.random() * gridSize);
51     // System.out.print(" try " + location);
52     int x = 0; // nth position in dotcom to place
53     success = true; // assume success
54     while (success && x < comSize) { // look for adjacent unused spots
55         if (grid[location] == 0) { // if not already used
56             coords[x++] = location; // save location
57             location += incr; // try 'next' adjacent
58             if (location >= gridSize) { // out of bounds - 'bottom'
59                 success = false; // failure
60             }
61             // out of bounds - right edge
62             if (x > 0 & (location % gridLength == 0)) {
63                 success = false; // failure
64             }

```

comCount: 1, 2, 3 순서로 증가

comCount가 홀수이면 세로로, 짝수이면 가로로 배치

attempts를 매번 1씩 증가시켜야 하므로 && 대신 & 를 사용함!

Startup을 배치 못하면 최대 200번 아래 반복

0 ~ 48 중 7로 가정

coords: 7, 14, 21

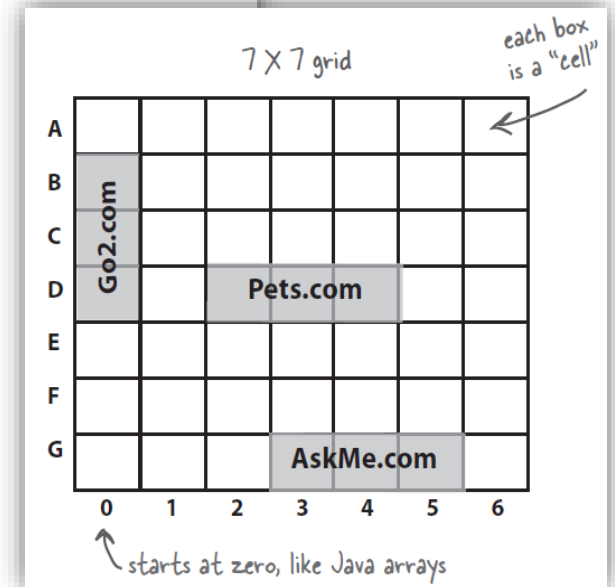
```

65         } else { // found already used location
66             // System.out.print(" used " + location);
67             success = false; // failure
68         }
69     }
70 } // end while
71
72 int x = 0; // turn good location into alpha coords
73 int row = 0;
74 int column = 0;
75 // System.out.println("Wn");
while (x < comSize) {
    grid[coords[x]] = 1; // 사용되는 칸으로 표시 : 'used'
    row = (int) (coords[x] / gridLength); // get row value
    column = coords[x] % gridLength; // get numeric column value
    temp = String.valueOf(alphabet.charAt(column)); // "abcdefg" 중 하나 선택
    alphaCells.add(temp.concat(Integer.toString(row))); // "0123456" 중 하나 선택
    x++;
}
return alphaCells; [b0, c0, d0]
}

```

coords[0] = 7의 경우
 row = (int) (7/7) = 1
 col = 7 % 7 = 0
 temp = "b"
 alphaCells = "b0"

coords[0] = 14의 경우
 row = (int) (14/7) = 2
 col = 14 % 7 = 0
 temp = "c"
 alphaCells = "c0"



"abcdefg" 중 하나 선택

"0123456" 중 하나 선택

* row, column 위치에 주의

StartupBust.java

```

1  package cse.oop2.ch06;
2
3  import java.util.ArrayList;
4
5  public class DotComBust {
6
7      private GameHelper helper = new GameHelper();
8      private ArrayList<DotCom> dotComsList = new ArrayList<DotCom>();
9      private int numOfGuesses = 0;
10
11     private void setUpGame() {
12         DotCom one = new DotCom();
13         one.setName("Pets.com");
14         DotCom two = new DotCom();
15         two.setName("eToys.com");
16         DotCom three = new DotCom();
17         three.setName("Go2.com");
18         dotComsList.add(one);
19         dotComsList.add(two);
20         dotComsList.add(three);
    
```

new ArrayList<>() 이 더 최신 문법!

```

22 System.out.println("Your goal is to sink three dot coms.");
23 System.out.println("Pets.com, eToys.com, Go2.com");
24 System.out.println("Try to sink them all in the fewest number of guesses");
25
26  for (DotCom dotComToSet : dotComsList) {
27     ArrayList<String> newLocation = helper.placeDotCom(3); [b0, c0, d0]
28     dotComToSet.setLocationCells(newLocation);
29 }
30 // dotComsList.forEach(e -> { functional operation 사용한 동일 코드
31 //     ArrayList<String> newLocation = helper.placeDotCom(3);
32 //     e.setLocationCells(newLocation);
33 // }); 더 간단한 표현: e.setLocationCells(helper.placeStartup(3));
34 }
35
36  private void startPlaying() { StartupsList에 있는 모든 Startup이 제거되어야만 끝남!
37     while (!dotComsList.isEmpty()) {
38         String userGuess = helper.getUserInput("Enter a guess"); 유효 값: "[a-g][0-6]"
39         checkUserGuess(userGuess);
40     }
41     finishGame();
42 }

```

```
44 private void checkUserGuess(String userGuess) {  
45     numOfGuesses++;  
46     String result = "miss";  
47  
48     for (DotCom dotComToTest : dotComsList) {  
49         result = dotComToTest.checkYourself(userGuess);  
50         if (result.equals("hit")) {  
51             break;  
52         }  
53         if (result.equals("kill")) {  
54             dotComsList.remove(dotComToTest);  
55             break;  
56         }  
57     }  
58     System.out.println(result);  
59 }
```

```
61  - private void finishGame() {  
62      System.out.println("All Dot Coms are dead! Your stock is now worthless");  
63      if (numOfGuesses <= 18) {  
64          System.out.println("It only took you " + numOfGuesses + " guesses");  
65          System.out.println("You got out before your options sank.");  
66      } else {  
67          System.out.println("Took you long enough. " + numOfGuesses + " guesses.");  
68          System.out.println("Fish are dancing with your options.");  
69      }  
70  }  
71  
72  - public static void main(String[] args) {  
73      DotComBust game = new DotComBust();  
74      game.setUpGame();  
75      game.startPlaying();  
76  }  
77  }
```

실행 결과

Your goal is to sink three dot coms.
 Pets.com, eToys.com, Go2.com
 Try to sink them all in the fewest number of guesses

Enter a guess a2	Enter a guess e3	Enter a guess e0
miss	hit	miss
Enter a guess a4	Enter a guess e4	Enter a guess c1
miss	miss	miss
Enter a guess b2	Enter a guess e1	Enter a guess e1
miss	kill	miss
Enter a guess c2	Enter a guess e2	Enter a guess c2
miss	miss	miss
Enter a guess c4	Enter a guess g2	Enter a guess c3
miss	hit	hit
Enter a guess d2	Enter a guess g3	Enter a guess d3
miss	hit	miss
Enter a guess d4	Enter a guess g4	Enter a guess b3
miss	kill	hit
Enter a guess e2	Enter a guess c0	Enter a guess a3
hit	miss	kill

All Dot Coms are dead! Your stock is now worthless
 Took you long enough. 27 guesses.
 Fish are dancing with your options.

6장 최종 코드: Startup.java

```
1 package cse.oop2.ch6;
2
3 import java.util.ArrayList;
4
5 public class Startup {
6     private ArrayList<String> locationCells;
7     private String name;
8
9     public void setLocationCells(ArrayList<String> loc) {
10         locationCells = loc;
11     }
12
13     public void setName(String n) {
14         name = n;
15     }
16 }
```

```
17 public String checkYourself(String userInput) {
18     String result = "miss";
19     int index = locationCells.indexOf(userInput);
20     if (index >= 0) {
21         locationCells.remove(index);
22         if (locationCells.isEmpty()) {
23             result = "kill";
24             System.out.println("Ouch! You sunk " + name + " : ( ");
25         } else {
26             result = "hit";
27         } // end if
28     } // end outer if
29     return result;
30 } // end method
31 } // close class
```

GameHelper.java

```
1 package cse.oop2.ch6;
2
3 import java.util.*;
4
5 public class GameHelper {
6     private static final String ALPHABET = "ABCDEFGH";
7     private static final int GRID_LENGTH = 7;
8     private static final int GRID_SIZE = 49;
9     private static final int MAX_ATTEMPTS = 200;
10
11     static final int HORIZONTAL_INCREMENT = 1;           // A better way to represent these two
12     static final int VERTICAL_INCREMENT = GRID_LENGTH;   // things is an enum (see Appendix B)
13
14     private final int[] grid = new int[GRID_SIZE];
15     private final Random random = new Random();
16
17     private int startupCount = 0;
```



```

19 public String getUserInput(String prompt) {
20     System.out.print(prompt + ": ");
21     Scanner scanner = new Scanner(System.in);
22     return scanner.nextLine().toLowerCase();
23 } //end getUserInput
24
25 public ArrayList<String> placeStartup(int startupSize) {
26     // holds index to grid (0 - 48)
27     int[] startupCoords = new int[startupSize];           // current candidate co-ordinates
28     int attempts = 0;                                     // current attempts counter
29     boolean success = false;                             // flag = found a good location?
30
31     startupCount++;                                       // nth Startup to place
32     int increment = getIncrement();                      // increment to try next location
33
34     while (!success & attempts++ < MAX_ATTEMPTS) {       // main search loop
35         int location = random.nextInt(GRID_SIZE);        // get random starting point
36
37         for (int i = 0; i < startupCoords.length; i++) { // create array of proposed coords
38             startupCoords[i] = location;                 // put current location in array
39             location += increment;                        // calculate the next location
40         }
41         System.out.println("Trying: " + Arrays.toString(startupCoords));

```

attempts를 매번 1씩 증가시켜야 하므로 && 대신 & 를 사용함!

```

43         if (startupFits(startupCoords, increment)) {           // startup fits on the grid?
44             success = coordsAvailable(startupCoords);          // ...and locations aren't taken?
45         }                                                       // end loop
46     }                                                           // end while
47
48     savePositionToGrid(startupCoords);                          // coords passed checks, save
49     ArrayList<String> alphaCells = convertCoordsToAlphaFormat(startupCoords);
50     System.out.println("Placed at: "+ alphaCells);
51     return alphaCells;
52 } //end placeStartup
53
54 boolean startupFits(int[] startupCoords, int increment) {
55     int finalLocation = startupCoords[startupCoords.length - 1];
56     if (increment == HORIZONTAL_INCREMENT) {
57         // check end is on same row as start
58         return calcRowFromIndex(startupCoords[0]) == calcRowFromIndex(finalLocation);
59     } else {
60         return finalLocation < GRID_SIZE;                      // check end isn't off the bottom
61     }
62 } //end startupFits

```

```
64 boolean coordsAvailable(int[] startupCoords) {
65     for (int coord : startupCoords) {                // check all potential positions
66         if (grid[coord] != 0) {                        // this position already taken
67             System.out.println("position: " + coord + " already taken.");
68             return false;                             // NO success
69         }
70     }
71     return true;                                     // there were no clashes, yay!
72 } //end coordsAvailable
73
74 void savePositionToGrid(int[] startupCoords) {
75     for (int index : startupCoords) {
76         grid[index] = 1;                             // mark grid position as 'used'
77     }
78 } //end savePositionToGrid
79
80 private ArrayList<String> convertCoordsToAlphaFormat(int[] startupCoords) {
81     ArrayList<String> alphaCells = new ArrayList<String>();
82     for (int index : startupCoords) {                 // for each grid coordinate
83         String alphaCoords = getAlphaCoordsFromIndex(index); // turn it into an "a0" style
84         alphaCells.add(alphaCoords);                  // add to a list
85     }
86     return alphaCells;                               // return the "a0"-style coords
87 } // end convertCoordsToAlphaFormat
```

```
89 String getAlphaCoordsFromIndex(int index) {
90     int row = calcRowFromIndex(index);           // get row value
91     int column = index % GRID_LENGTH;           // get numeric column value
92
93     String letter = ALPHABET.substring(column, column + 1); // convert to letter
94     return letter + row;
95 } // end getAlphaCoordsFromIndex
96
97 private int calcRowFromIndex(int index) {
98     return index / GRID_LENGTH;
99 } // end calcRowFromIndex
100
101 private int getIncrement() {
102     if (startupCount % 2 == 0) {                 // if EVEN Startup
103         return HORIZONTAL_INCREMENT;             // place horizontally
104     } else {                                     // else ODD
105         return VERTICAL_INCREMENT;               // place vertically
106     }
107 } //end getIncrement
108 } //end class
```

StartupBust.java

```
1 package cse.oop2.ch6;
2
3 import java.util.ArrayList;
4
5 public class StartupBust {
6
7     private GameHelper helper = new GameHelper();
8     private ArrayList<Startup> startups = new ArrayList<Startup>();
9     private int numOfGuesses = 0;
10
11     private void setUpGame() {
12         // first make some Startups and give them locations
13         Startup one = new Startup();
14         one.setName("poniez");
15         Startup two = new Startup();
16         two.setName("hacqi");
17         Startup three = new Startup();
18         three.setName("cabista");
19         startups.add(one);
20         startups.add(two);
21         startups.add(three);
```

```
23     System.out.println("Your goal is to sink three Startups.");
24     System.out.println("poniez, hacqi, cabista");
25     System.out.println("Try to sink them all in the fewest number of guesses");
26
27     for (Startup startup : startups) {
28         ArrayList<String> newLocation = helper.placeStartup(3);
29         startup.setLocationCells(newLocation);
30     } // close for loop
31 } // close setUpGame method
32
33 private void startPlaying() {
34     while (!startups.isEmpty()) {
35         String userGuess = helper.getUserInput("Enter a guess");
36         checkUserGuess(userGuess);
37     } // close while
38     finishGame();
39 } // close startPlaying method
```

```
41 private void checkUserGuess(String userGuess) {
42     numOfGuesses++;
43     String result = "miss"; // assume a miss until told otherwise
44
45     for (Startup startupToTest : startups) {
46         result = startupToTest.checkYourself(userGuess);
47
48         if (result.equals("hit")) {
49             break;
50         }
51         if (result.equals("kill")) {
52             startups.remove(startupToTest); // he's gone
53             break;
54         }
55     } // close for
56
57     System.out.println(result);
58 } // close method
```

```
60     private void finishGame() {
61         System.out.println("All Startups are dead! Your stock is now worthless");
62         if (numOfGuesses <= 18) {
63             System.out.println("It only took you " + numOfGuesses + " guesses.");
64             System.out.println("You got out before your options sank.");
65         } else {
66             System.out.println("Took you long enough. " + numOfGuesses + " guesses.");
67             System.out.println("Fish are dancing with your options");
68         }
69     } // close method
70
71     public static void main(String[] args) {
72         StartupBust game = new StartupBust();
73         game.setUpGame();
74         game.startPlaying();
75     } // close method
76 }
```


부울 연산자

- &&, ||
 - &&는 양쪽에 있는 부울 표현식이 모두 참인 경우에, ||는 양쪽에 있는 부울 표현식 중 하나 이상이 참인 경우에 참이 됩니다.

```
if (price >= 300 && price < 400) { }
```

```
if (brand.equals("A") || brand.equals("B")) { }
```

- !=, !
 - 논리적 NOT을 의미하는 연산자.

```
if (model != 2000) { }
```

```
if (!brand.equals("X")) { }
```

부울 연산자

- 단락 연산자(short circuit operator: &&, ||)

```
if (price >= 300 && price < 400) { }
```

```
if (brand.equals("A") || brand.equals("B")) { }
```

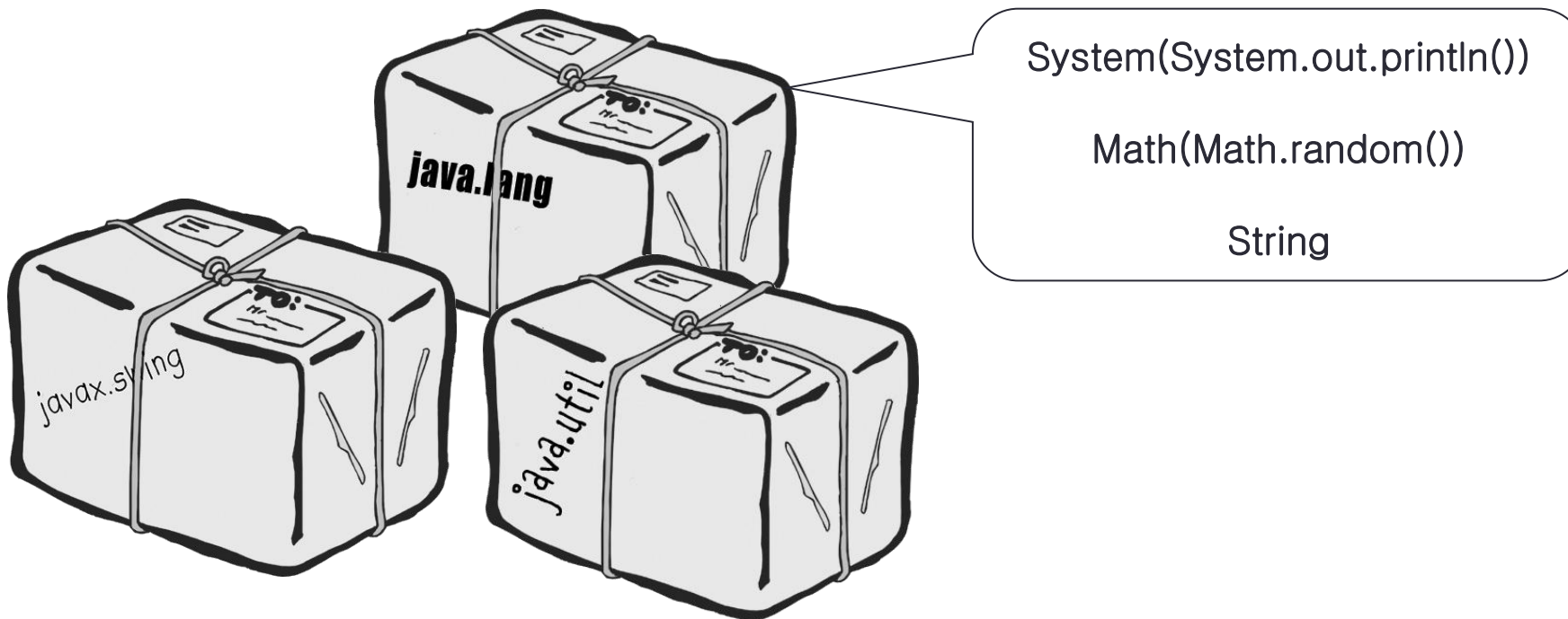
```
if (refVar != null && refVar.isValidType()) { }
```

- 비단락 연산자(&, |) : non-short-circuit logic operators

- 반드시 양쪽 표현식을 모두 실행시키고 싶은 경우에는 비단락 연산자를 쓰면 됩니다.
- 비트 별 연산에도 사용됨.

라이브러리 (자바 API)

- 자바 API에서 클래스는 패키지 단위로 묶여 있습니다.
 - API에 들어있는 클래스를 사용하려면 그 클래스가 어떤 패키지에 들어있는지 알아야 합니다.



클래스의 전체 이름

- 클래스를 사용하려면 클래스의 전체 이름을 알아야 합니다.
 - 예외: java.lang 패키지에 들어있는 클래스
- ArrayList의 전체 이름
 - java.util.ArrayList

클래스의 이름을 알려주는 방법

A. import 선언문을 씁니다.

```
import java.util.ArrayList; // 개별 클래스나 인터페이스의 import
```

```
Import java.util.*; // 패키지 내 모든 클래스와 인터페이스의 import
```

B. 일일이 입력합니다.

```
java.util.ArrayList<Dog> list = new java.util.ArrayList<>();
```

```
public void go(java.util.ArrayList<Dog> list) { }
```

```
public java.util.ArrayList<Dog> foo() { }
```

Q&A

- 왜 꼭 전체 이름이 필요하죠? 패키지의 목적이 그것 뿐인가요?
 - 프로젝트 또는 라이브러리를 더 용이하게 편성할 수 있습니다.
 - 이름 영역을 지정해서 같은 이름 때문에 충돌이 생기는 것을 방지할 수 있습니다.
 - 어느 정도의 보안 기능을 제공할 수 있습니다.
- *cf.* FQN: Fully Qualified Name

- 전체 이름을 사용하면 어떤 장점이 있습니까? 클래스명뿐 아니라 패키지명까지 똑같이 붙이면 어떻게 해야 하나요?
- 자바에는 이런 문제가 생기는 것을 방지하기 위한 명명 규칙 같은 것이 있습니다. 이 규칙만 지키면 별로 문제가 생기지 않겠죠.

핵심정리

- ArrayList는 자바 API에 포함되어있는 클래스입니다.
- ArrayList에 뭔가를 집어넣을 때는 `add()`를 쓰면 됩니다.
- ArrayList에서 뭔가를 제거할 때는 `remove()`를 쓰면 됩니다.
- ArrayList에 들어있는 어떤 것의 위치를 알고 싶다면(또는 들어있는지 알고 싶다면) `indexOf()`를 쓰면 됩니다.
- ArrayList가 비어 있는지 확인할 때는 `isEmpty()`를 쓰면 됩니다.
- ArrayList의 크기(원소의 개수)를 알고 싶다면 `size()` 메서드를 쓰면 됩니다.
- 일반 배열의 길이(원소의 개수)를 알고 싶을 때는 `length` 변수를 쓰면 됩니다.

- ArrayList는 필요에 따라 그 크기가 동적으로 바뀝니다.
- ArrayList를 선언할 때는 거기에 저장할 객체 유형을 매개변수로 지정합니다.
- ArrayList에는 객체만 저장할 수 있지만 원시 유형을 집어넣어도 컴파일러에서 자동으로 처리해줍니다. (래퍼 클래스, auto-boxing, auto-unboxing)
- 클래스는 패키지 단위로 묶입니다.
- 클래스에는 패키지 명과 클래스 명을 합쳐서 만든 전체 이름(FQN)이 있습니다:

ArrayList → java.util.ArrayList

- java.lang을 제외한 다른 패키지에 들어있는 클래스를 사용하려면 자바에 클래스의 전체 이름을 알려줘야 합니다.

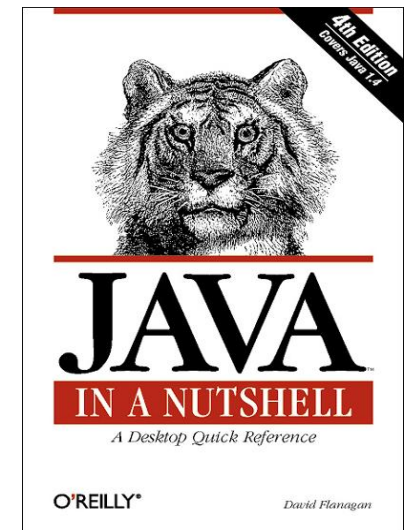
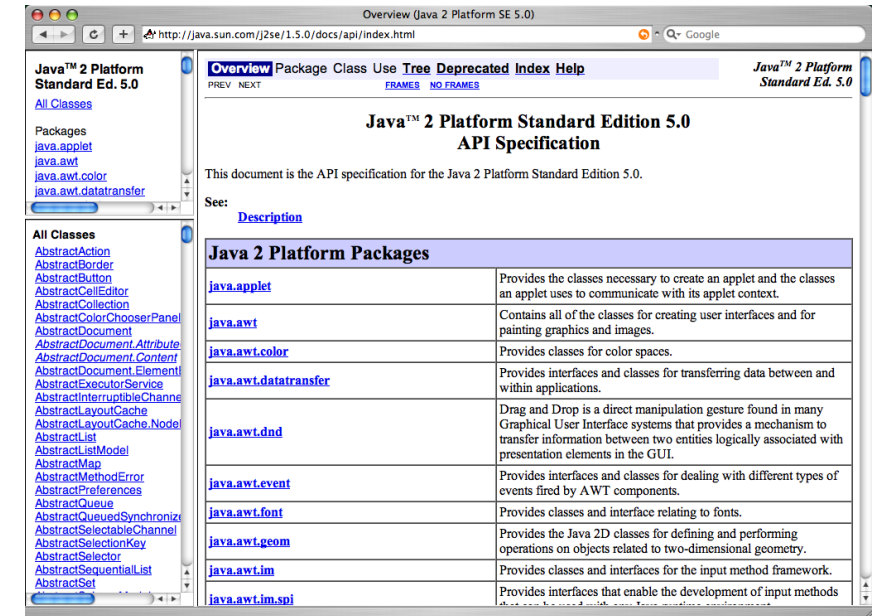
- import를 사용하면 클래스가 커지나요? 그렇게 불러온 클래스나 패키지가 내가 만든 코드에 추가되어 같이 컴파일되나요?
 - C를 쓰고 있었다면 그렇게 생각할지도 모르겠습니다만, 자바에서는 그렇지 않습니다. import를 쓴다고 해서 코드 분량이 늘어나거나 느려지는 일은 없습니다. import는 단순히 클래스의 전체 이름을 알려주기 위한 도구에 불과합니다.

- 그러면 String이나 System 같은 클래스는 왜 import 선언문 없이 써도 괜찮은가요?
 - **java.lang** 패키지는 **import** 선언문을 쓰지 않아도 자동으로 들어갑니다. 예를 들어 String, System 같은 클래스는 따로 import 선언문을 쓰지 않아도 전체 이름이 아닌 클래스 이름만으로 사용할 수 있습니다.

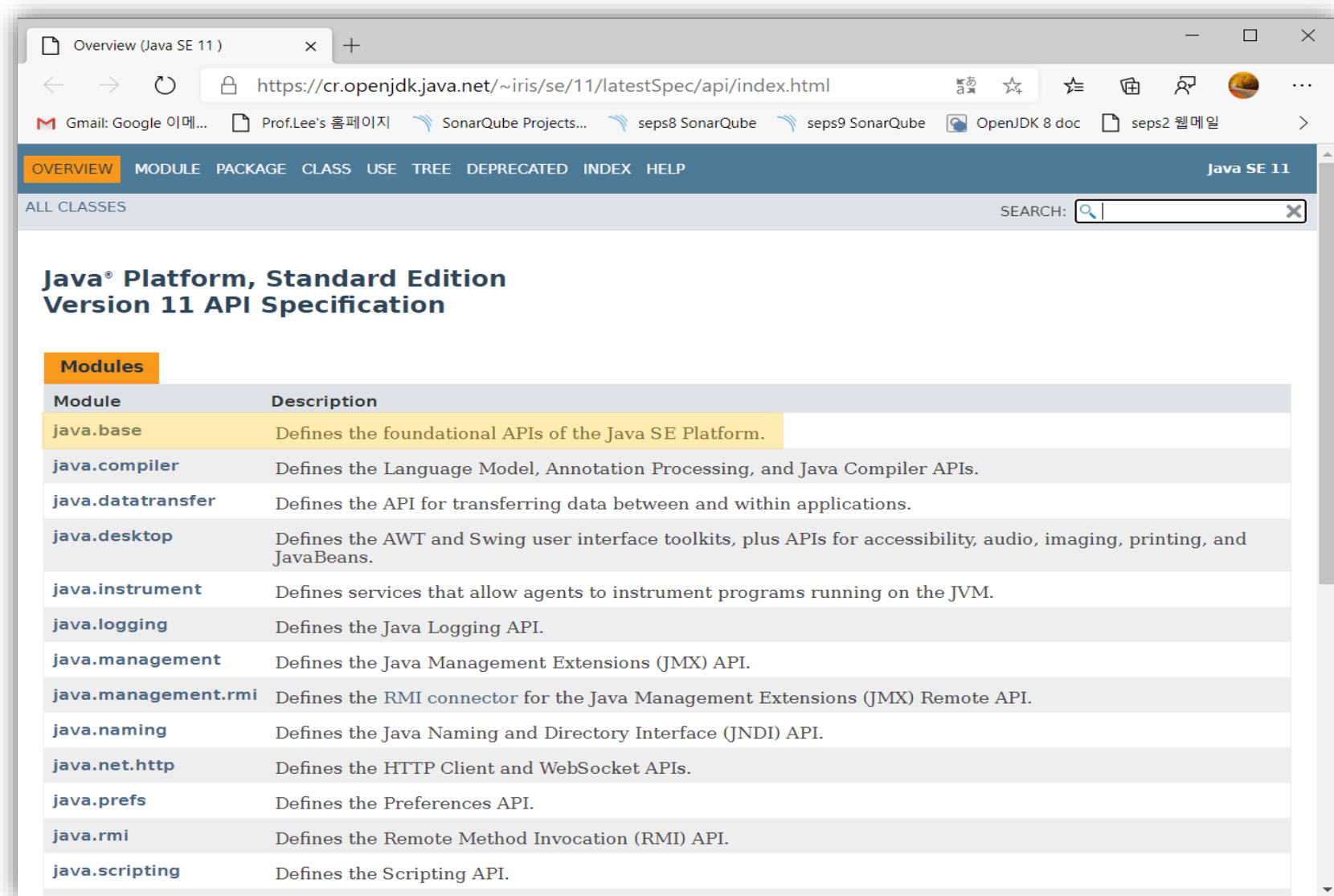
- 내가 만든 클래스도 패키지에 집어넣어야 하나요? 어떻게 하죠? 정말 그렇게 할 수 있나요?
- 아마 실제 프로그래밍을 할 때는 항상 클래스를 패키지에 넣어야 할 것입니다. 자세한 내용은 나중에 패키징과 관련된 장에서 알아보도록 하겠습니다.

API 활용 방법

- 라이브러리에는 어떤 클래스가 들어있을까?
 - 책을 훑어봅니다.
- 클래스를 찾았을 때 그것으로 무엇을 할 수 있을지 어떻게 알 수 있을까?
 - HTML API 문서를 활용합니다.
 - <https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/index.html>
 - <https://docs.oracle.com/en/java/javase/11/docs/api/index.html>



HTML API 문서를 활용하는 방법



The screenshot shows the Java SE 11 API Specification website. The browser address bar displays the URL: <https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/index.html>. The page title is "Java® Platform, Standard Edition Version 11 API Specification". The "Modules" section is highlighted, showing a table of modules and their descriptions.

Module	Description
<code>java.base</code>	Defines the foundational APIs of the Java SE Platform.
<code>java.compiler</code>	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
<code>java.datatransfer</code>	Defines the API for transferring data between and within applications.
<code>java.desktop</code>	Defines the AWT and Swing user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
<code>java.instrument</code>	Defines services that allow agents to instrument programs running on the JVM.
<code>java.logging</code>	Defines the Java Logging API.
<code>java.management</code>	Defines the Java Management Extensions (JMX) API.
<code>java.management.rmi</code>	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.
<code>java.naming</code>	Defines the Java Naming and Directory Interface (JNDI) API.
<code>java.net.http</code>	Defines the HTTP Client and WebSocket APIs.
<code>java.prefs</code>	Defines the Preferences API.
<code>java.rmi</code>	Defines the Remote Method Invocation (RMI) API.
<code>java.scripting</code>	Defines the Scripting API.

숙제

- 본문을 다시 한 번 꼼꼼히 읽어봅시다.
- 6장 본문 및 맨 뒤에 나와있는 연습문제를 직접 풀어봅시다.
- 6장에 나와있는 닷컴 가라앉히기 게임을 직접 타이핑해서 컴파일 및 실행시켜봅시다.
- JAVA 21 API 문서를 인터넷에서 찾아 그동안 배울 클래스와 메서드를 찾아 봅시다. JAVA 11 API 문서는 시험칠 때 제공되므로 클래스와 메서드 찾는 방법을 잘 알아 두어야 합니다.